Circular Trading Detection using Node2Vec

Vijay Varma K AI20BTECH11012
Sachin Karumachi AI20BTECH11013
Abhiroop Ch AI20BTECH11005
Jeevan S CS20BTECH11047
Jatin Kumar CS19BTECH11036

May 4, 2023

Abstract

Node2Vec is a powerful algorithm for generating low-dimensional vector representations, or embedding, of nodes in a network. It has become a popular approach for various network analysis tasks such as link prediction, node classification, and clustering. The algorithm is based on the idea of learning continuous representations of nodes by preserving both the structural information and the network topology. By leveraging a biased random walk strategy, Node2Vec can capture both the local and global neighborhood structure of nodes in the network and create embeddings that reflect the semantic meaning of nodes.

1 Problem statement

We were given a large set of data with information on dealers and we need to analyse that data to find the dealers involved in circular trading.

2 What is Circular Trading?

Circular trading is a fraudulent financial activity in which a group of companies or individuals create a cycle of transactions to artificially inflate their revenue, profits, and share prices. The scheme typically involves several companies, each owned or controlled by the same group of people, that engage in circular sales and purchases of goods or services to one another, often at inflated prices or with fictitious invoices. The purpose of circular trading is to create the appearance of legitimate business activities and financial performance, attract investors or lenders, and ultimately enrich the organizers of the scheme.

3 Motivation

Many graph analyzing algorithms don't give much information when the graphs are complex or large, but Node2Vec leverages random walks and generates most informative embeddings out of all the graph analyzing algorithms and is still efficient for large and complex graph structures.

4 Data set description

The given Dataset consists of 3 columns "Seller ID", "Buyer ID" and "Value"

- Seller ID: Unique ID given to a dealer, this column consists of dealers selling goods (This is of type INT)
- Buyer ID: Unique ID given to a dealer, this column consists of dealers buying goods from seller (This is of type INT)
- Value: It indicates the value of transaction between seller and buyer (This is of type FLOAT)

The objective is to identify malicious dealers using this data.

5 Algorithm and its implementation

5.1 Algorithm

5.1.1 Node2Vec

1. Construct an undirected graph representing the network and assign weights to the edges based on their strength or importance.
2. For each node in the network, perform a random walk of a fixed length, by selecting the next node to visit based on a probability distribution that balances between exploring new nodes and revisiting previously visited nodes. The probability distribution is controlled by two parameters, called "p" and "q", which influence the tendency of the random walk to stay in the local neighborhood or to explore farther away.
3. Generate a sequence of nodes visited by the random walk for each starting node, and use these sequences to create a corpus of text data.
4. Apply a word embedding algorithm such as Word2Vec or GloVe to the corpus of text data, to learn low-dimensional vector representations of each node in the network.
5. Use the learned embeddings to perform various network analysis tasks such as link prediction, node classification, and clustering.

5.1.2 tSNE

1. Compute pairwise similarities between data points in high-dimensional space, using a Gaussian kernel or other similarity measure.
2. Initialize the embedding of the data points in a low-dimensional space, randomly or using some other initialization method.
3. Compute the similarities between the embedded points in the low-dimensional space, using a Student's t-distribution.
4. Optimize the embedding by minimizing the Kullback-Leibler divergence between the pairwise similarities of the high-dimensional data and the pairwise similarities of the embedded points in the low-dimensional space. This is done using gradient descent or other optimization methods.
5. Repeat steps 3 and 4 until convergence or a specified number of iterations.

5.1.3 DBSCAN

1. Define a distance metric between data points, such as Euclidean distance or Manhattan distance.
2. Choose two parameters: epsilon (ϵ), which defines the maximum distance between two points to be considered as neighbors, and minPoints, which specifies the minimum number of points required to form a dense region.
3. Select a random unvisited point from the dataset and find its ϵ-neighborhood, i.e., all the points within a distance of ϵ from the selected point.
4. If the number of points in the ϵ-neighborhood is greater than or equal to minPoints, then form a cluster around the selected point, including all points in its ϵ-neighborhood that also have at least minPoints points in their ϵ-neighborhood.
5. Repeat steps 3-4 for all unvisited points until all points have been visited.
6. Any unvisited point that is not part of any cluster is considered an outlier or noise.

5.2 Drawbacks of the used algorithms

5.2.1 Node2Vec

1. *Computationally expensive:* Node2Vec involves a large number of random walks, which can be computationally expensive and time-consuming, especially for large and dense networks. This can limit the scalability of the algorithm, and require significant computational resources.

2. *Sensitivity to hyperparameters:* Node2Vec has several hyperparameters, such as the length of random walks and the weights of the two types of biased walks, that need to be tuned to achieve optimal results. However, the optimal values of these hyperparameters may depend on the specific network and the task at hand, making it challenging to find the best hyperparameters in practice.

3. *Limited interpretability:* While Node2Vec can learn high-quality node embeddings that capture the structural and semantic properties of the network, the embeddings themselves may be difficult to interpret and understand. This can limit the ability to gain insights into the network and to use the embeddings for downstream tasks.

4. *Sensitivity to network topology:* Node2Vec assumes that the network is locally homogeneous and that nodes with similar neighborhoods should have similar embeddings. However, this assumption may not hold for all networks, especially for networks with complex and diverse topologies. This can result in suboptimal embeddings that do not reflect the true structure of the network.

5.2.2 tSNE

1. *Computationally expensive:* t-SNE is computationally expensive and can take a long time to run, especially for large datasets with many dimensions. This can limit its applicability to datasets that are too large or complex for practical use.

2. *Non-convex optimization:* t-SNE is a non-convex optimization problem, which means that it can get stuck in local optima and may not find the global optimum. This can result in different embeddings for the same dataset with different initializations.

3. *Sensitivity to hyperparameters:* t-SNE has several hyperparameters, such as the perplexity and learning rate, that need to be carefully tuned to achieve optimal results. However, the optimal values of these hyperparameters may depend on the specific dataset and the task at hand, making it challenging to find the best hyperparameters in practice.

4. *Interpretability:* While t-SNE can produce high-quality visualizations that reveal the underlying structure of the data, the embeddings themselves may be difficult to interpret and understand. This can limit the ability to gain insights into the data and to use the embeddings for downstream tasks.

5.2.3 DBSCAN

1. *Sensitivity to hyperparameters:* DBSCAN has several hyperparameters, such as the radius of the neighborhood and the minimum number of points required to form a cluster, that need to be carefully chosen to achieve optimal results. However, the optimal values of these hyperparameters may depend on the specific dataset and the task at hand, making it challenging to find the best hyperparameters in practice.

2. *Scalability:* DBSCAN has a quadratic time complexity with respect to the number of data points, which means that it can be computationally expensive and time-consuming for large datasets. However, there are variants of DBSCAN that have been developed to improve its scalability.

3. *Difficulty with clusters of varying densities and shapes:* DBSCAN assumes that clusters are dense regions of data points separated by areas of lower density. This means that it may have difficulty identifying clusters that have varying densities and shapes, such as elongated or irregularly shaped clusters.

4. *Sensitivity to the distance metric:* DBSCAN's performance can be sensitive to the choice of distance metric used to measure the similarity between data points. Choosing an appropriate distance metric that captures the relevant characteristics of the data can be challenging in practice.

5.3 Algorithm Implementation

1. Using the dataset, create a adjacent list for building a Directed graph G Directed from "Seller ID" to "Buyer ID" with weight of an edge as "Value".

2. Now create a undirected graph $G_{weighted}$ using the nodes of G and calculate number of two-cycles and three triangles between two nodes u and it's neighbor and update the weight of edge $(u, neighbor(u))$ by number of two-cycles + number of three triangles.

3. Taking graph $G_{weighted}$ as input for Node2Vec algorithm.

   (a) *Hyperparameters taken*

      i. $dimensions = 30$
      ii. $walklength = 50$
      iii. $numwalks = 100$
      iv. $p = 0.7$
      v. $q = 0.3$
      vi. $window = 5$

   (b) This Node2Vec algorithm return embeddings for each node in a vector form of 30 dimensions (There are 799 nodes in the graph).

4. We take these embeddings as input for tSNE for dimensionality reduction.

   (a) *Hyperparameters taken*

      i. no of components= 2
      ii. $perplexity = 30$
      iii. $learningrate = 300$

   (b) This reduces the dimension of each data point from 30 to 2, now number of dimensions of each node is 2.

5. Now the output from the tSNE is our new embeddings now we cluster these data points using DBSCAN.

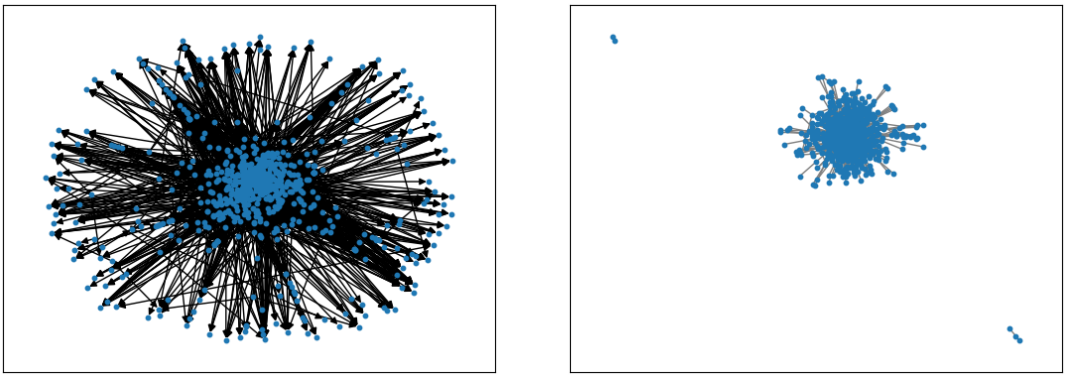
   (a) *Hyperparameters taken*

      i. $eps = 0.2$
      ii. min samples= 5

   (b) This clusters our data points depending on distance between our data points in our new embeddings.

6. We can find the outliers and distant groups involving circular trading by the end output from the clustering algorithm.

6 Results



(a) Directed Graph (b) Undirected Weighted Graph

Figure 1: Graphs

The Total Number of Nodes in the above Undirected Weighted Graph are 799.
The Number of Edges in the above Undirected Weighted Graph are 5040.

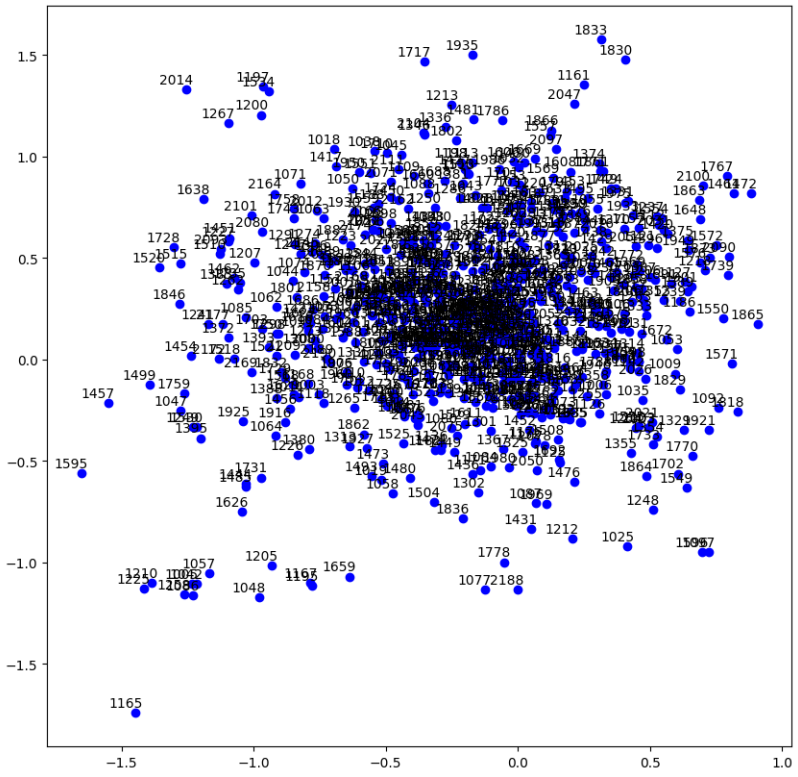


Figure 2: Node2Vec Embeddings

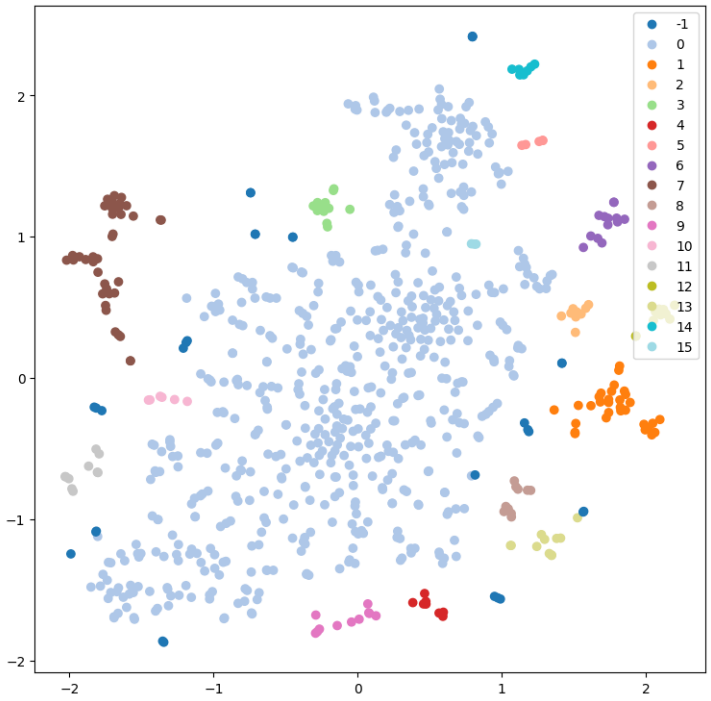


Figure 3: Clusters after applying tSNE and DBSCAN

They are many clusters formed in the above Plot.

- Cluster -1 has 31 Nodes. These are Outliers as per DBSCAN.
- Cluster 0 has 584 Nodes.
- Cluster 1 has 31 Nodes.
- Cluster 2 has 9 Nodes. They are ['1139', '2127', '1190', '1743', '1671', '1440', '1932', '2027', '2162'].
- Cluster 3 has 12 Nodes.
- Cluster 4 has 8 Nodes. They are ['1426', '1029', '1363', '1716', '1191', '1927', '2150', '1421'].
- Cluster 5 has 5 Nodes. They are ['1145', '1410', '1304', '1588', '1069'].
- Cluster 6 has 13 Nodes.
- Cluster 7 has 38 Nodes.
- Cluster 8 has 10 Nodes. They are ['1111', '1187', '1183', '1374', '2097', '1115', '1871', '1102', '1543', '1942'].
- Cluster 9 has 11 Nodes. They are ['1405', '1218', '1121', '1691', '2028', '1566', '1295', '1606', '1648', '1740', '1811'].
- Cluster 10 has 6 Nodes. They are ['1282', '1152', '1961', '1550', '1575', '1571'].
- Cluster 11 has 10 Nodes. They are ['1196', '1385', '2084', '1380', '1585', '1352', '2012', '1176', '1325', '1485'].
- Cluster 12 has 10 Nodes. They are ['1085', '1062', '1456', '1274', '1241', '1846', '1462', '1735', '1383', '1515'].
- Cluster 13 has 11 Nodes. They are ['1157', '1061', '1434', '1765', '1777', '1519', '1294', '1293', '1861', '1763', '1554'].
- Cluster 14 has 7 Nodes. They are ['1248', '1255', '1549', '1596', '1025', '2021', '1097'].
- Cluster 15 has 3 Nodes. They are ['2008', '1333', '1800'].

7 Conclusion

- From the above results, we can tell that Circular Trading is happening among the Traders in the above Clusters with each other.
- Node2Vec algorithm is used to analyze complex graph data, here we have taken data of Iron dealers to find the possibility of circular trading between the dealers. It is a powerful algorithm for generating high-quality node embeddings in graphs.
- Using random walks, it can capture both the structural and semantic properties of the graph and produce embeddings that are useful for a variety of downstream tasks.
- Additionally, Node2Vec offers flexibility in choosing the hyperparameters to tune for specific applications, and can be used with different graph structures and sizes.
- To visualize the the graph embeddings generated by Node2Vec we use tSNE or PCA to reduce the dimensions of embeddings and visualize these embeddings and finally we cluster these data points using DBSCAN, K-means clustering to find the group of dealers have a possibility of involving in circular trading

8 References

- Node2Vec -link
- Word2Vec paper -link
- ChatGpt
- Lecture notes of tSNE and DBScan.